Java mapping for pragmatic programmers

October 18, 2013

ENGINEERING BUSINESS PERFORMANCE

Java mapping for pragmatic programmers

Scala Edition

October 18, 2013

LUXOFT
ENGINEERING BUSINESS PERFORMANCE

Disclaimer

Disclaimer

- I'm not a Scala expert.

Disclaimer

- I'm not a Scala expert.
- Actually, I do not know Scala at all.

Disclaimer

- I'm not a Scala expert.
- Actually, I do not know Scala at all.
- Lets discuss today not implementations, but ideas staying behind them.

Agenda

1. Express Scala Course
2. Parser combinators (xRools)
3. JAXB plugin
4. Scala Macros (yajom)
5. Q&A, Discussion

Express Scala Course

- Not a full Scala description
- Not a functional programming course
- The main task of “course” is to help read Scala code, not write
- Java8 is recommended background

Java

Scala

```java
final String a = "a";
String b = "b";

void setC(C c) {
  this.c = c;
}

int inc(int a) {
  return a + 1;
}
```

Java

```java
final String a = "a";
String b = "b";

void setC(C c) {
  this.c = c;
}

int inc(int a) {
  return a + 1;
}
```

Scala

```scala
val a : String = "a"
var b = "b"

def setC(c:C):Unit =
{
  this.c = c;
}

def inc(a:Int) = a+1
```

Java

Scala

```
List<A> list =
  new ArrayList<>();

int l = list.size();

int[] arr;
int x = arr[5];

list.filter(
  a -> a.hasB()
).map(a -> new B(a))
```

Java

```java
List<A> list =
  new ArrayList<>();

int l = list.size();

int[] arr;
int x = arr[5];

list.filter(
  a -> a.hasB()
).map(a -> new B(a))
```

Scala

```scala
val list : List[A] =
  new ArrayList[A]()

val l = list.size

var arr : Array[Int]
val x: Int = arr(5)

list
.filter(_.hasB)
.map(a => new B(a))
```

xRools

- Written by Scala experts from Kiev
- XPath like DSL
- Extendable by Scala code
- XML oriented

xRools

Talk is cheap. Show me the code.

xRools Pros

- Scala

xRools Pros

- Scala
- Non-developers write rules

xRools Pros

- Scala
- Non-developers write rules
- Dashboard

xRools Cons

- Scala

xRools Cons

- Scala
- Non-developers write rules

xRools Cons

- Scala
- Non-developers write rules
- No compile-time checks

xRools Cons

- Scala
- Non-developers write rules
- No compile-time checks
- Uses reflection

xRools Cons

- Scala
- Non-developers write rules
- No compile-time checks
- Uses reflection
- No real IDE support

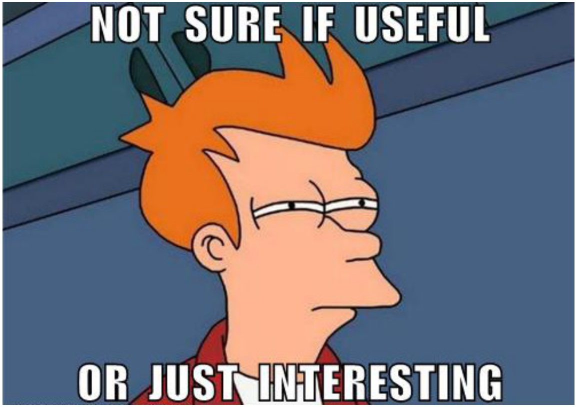
xRools Cons

- Scala
- Non-developers write rules
- No compile-time checks
- Uses reflection
- No real IDE support
- Functions defined away from its only usage

xRools Cons

- Scala
- Non-developers write rules
- No compile-time checks
- Uses reflection
- No real IDE support
- Functions defined away from its only usage
- Debugging?

xRools



JAXB plugin

- Written by Sergey Armensky
  (idea of Andrey Vytnov?)
- Plain Java (new method added for each property)
- Will be soon in production
- Have nothing to do with Scala

JAXB plugin

Talk is cheap. Show me the code.

JAXB plugin Pros

- Pure Java

JAXB plugin Pros

- Pure Java
- Simple and clean

JAXB plugin Pros

- Pure Java
- Simple and clean
- Simplify some code

JAXB plugin Cons

- Only for JAXB generated classes

JAXB plugin Cons

- Only for JAXB generated classes
- Simplify only some code

JAXB plugin Cons

- Only for JAXB generated classes
- Simplify only some code
- Not declarative enough

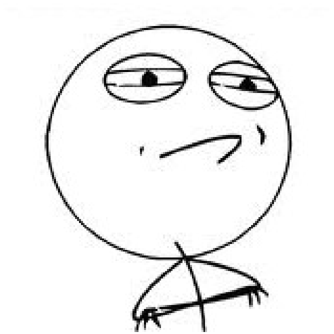
JAXB plugin Cons

- Only for JAXB generated classes
- Simplify only some code
- Not declarative enough
- Not mapper at all

JAXB Plugin

Can we do better?

Can we do better?



CHALLENGE ACCEPTED

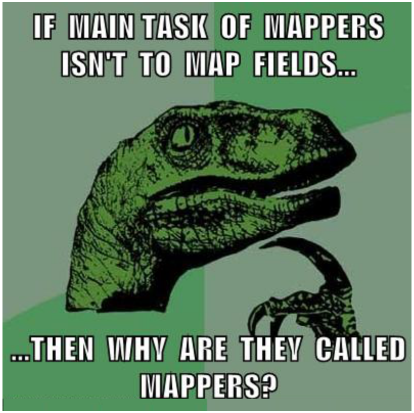
Scala related features

- Implicits of all kinds
- Point-free style
- Scala virtualized
- Macros

Yajom

- Yet Another Java Object Mapper
- Based on implicits and Scala macros
- All reflection is done in compile-time
- Easy to configure, extend or derive
  (having no clue about Scala Macros at all)

Philosoraptor


IF MAIN TASK OF MAPPERS ISN'T TO MAP FIELDS...
...THEN WHY ARE THEY CALLED MAPPERS?

Java with JAXB plugin

```
A a = from.getA();
if (from.isVerySpecial()) {
  B b = defaultB;
  if (a != null) {
    B realB = a.getB();
    if (realB != null)
      b = realB;
  }
  to.c().setB(b);
}
```

Option-based yajom

```
val option = {
  val a = from.getA
  if (from.isVerySpecial) {
    var b = defaultB
    if (a != null) {
      val realB = a.getB
      if (realB != null)
        b = realB
    }
    Some(b)
  } else None
}
yajomOption(to.getC.setB)(option)
```

Option-based yajom

```
yajomOption(to.getC.setB) {
  val a = from.getA
  if (from.isVerySpecial) {
    var b = defaultB
    if (a != null) {
      val realB = a.getB
      if (realB != null)
        b = realB
    }
    Some(b)
  } else
    None
}
```

Option-based yajom with maybe

```
yajomOption(to.getC.setB) {
  if (from.isVerySpecial) {
    val b = maybe(from.getA.getB)
    Some(b.getOrElse(defaultB))
  } else
    None
}
```

Yajom

Talk is cheap. Show me the code.

Yajom Pros

- Scala

Yajom Pros

- Scala
- Compile-time bulletproof

Yajom Pros

- Scala
- Compile-time bulletproof
- IDE support

Yajom Pros

- Scala
- Compile-time bulletproof
- IDE support
- Works with any classes without any changes

Yajom Pros

- Scala
- Compile-time bulletproof
- IDE support
- Works with any classes without any changes
- No problem with debugging

Yajom Cons

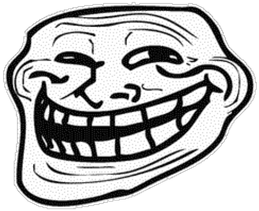
- Scala

Yajom Cons

- Scala
- Maybe not mature enough

Yajom Cons

- Scala
- Maybe not mature enough



Yajom

https://github.com/gark87/yajom

Thank You!